UNCLASSIFIED

AFIT/EN-TR-91-10

Air Force Institute of Technology

Using Ada Bindings to the X Window System

Gary W. Klabunde
Capt, USAF

Mark A. Roth
Maj, USAF

30 December 1991

Approved for public release; distribution unlimited

Using Ada Bindings to the X Window System

Gary W. Klabunde
Mark A. Roth

December 30, 1991

1 Introduction

The graphical user interface has been accepted as being one of the most important parts of user interactive software projects. Until recently, however, the design and implementation of such interfaces in Ada was a long and difficult process. A solution to this problem was found with the introduction of the X Window System in the mid 1980's. These C libraries make it relatively easy to construct sophisticated user interfaces. In the last couple of years, several organizations have developed bindings to, or implementations of, the X Window System software for use in Ada programs. Two of the more significant accomplishments were the bindings developed by the Science Applications International Corporation (SAIC) [1, 2] and those developed by the Boeing Aerospace Corporation [3].

In 1987, SAIC developed Ada bindings to X11R4. Their work was performed under a Software Technology for Adaptable Reliable Systems (STARS) foundation contract, and is therefore in the public domain. Stephen Hyland gives an overview of these bindings in [1].

In this paper we present a detailed example of how to use the SAIC and Boeing bindings to interface an Ada program to X. We discuss issues involving SAIC/Boeing binding compatibility, pixmap creation and use, connection with the X server, creation of several different widgets, and adding appropriate callback procedures. We give a more general description of the issues involved in building Ada/X interfaces in [5].

2 Ada Bindings to Xt Intrinsics and Motif

In a manner similar to that used by SAIC, the Boeing Corporation recently developed Ada bindings to a large subset of the Xt Intrinsics and the Motif widget set. Their code also provides access to a very limited subset of Xlib functions and data types. Like the SAIC code, Boeing's effort was sponsored by a STARS contract. For the most part, the subroutine names and parameter lists closely mirror the actual C routines. Also, Boeing added a few subprograms to assist in the building of some commonly used parameter lists. The bindings require the Verdix Ada Development System (VADS) version 5.5 or higher to execute. While the documentation on the software is relatively sparse, it does indicate which modules would require changes in order to port the bindings to other systems.

The Boeing bindings consist of several packages, each containing bindings to a particular portion of the X Window System. Package XLIB exports types for such things as *Drawable*, *Screen*, *Window*, and *Display*. It also contains bindings to less than a dozen Xlib functions. The X_TOOLKIT_INTRINSICS_OSF package exports the *Widget* and *Pixel* types among others. It contains bindings to most of the major Xt Intrinsics functions. The two remaining packages, XM_WIDGET_SET and MOTIF_RESOURCE_MANAGER, contain bindings for Motif v1.0 and the MRM, respectively.

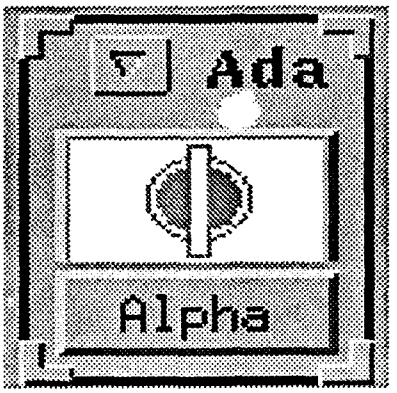

Figure 1: Saber Airbase Representation

3 Combining the SAIC and Boeing Bindings

The SAIC and Boeing bindings were successfully used at the Air Force Institute of Technology (AFIT) to create a graphical user interface for the Saber computerized wargame [4]. Saber is a theater level wargame developed for the Air Force Wargaming Center. While some user interfaces can be developed using only the Xt Intrinsics and a widget set, many also require some of the lower level functions provided by Xlib. For the Saber user interface, the SAIC bindings were needed to construct the graphics contexts used to display symbols for such things as airbases, aircraft missions, and ground units. However, making the few calls to the SAIC code was not straightforward because of different types used by the two sets of bindings.

The code that follows was extracted from the Saber user interface. It illustrates how to connect with the X server using the bindings and creates a symbol representing an airbase (see Figure 1). The symbol consists of a form widget that encloses two pushbutton widgets arranged vertically. The top pushbutton is drawn with a pixmap that depicts the symbol for an airbase. Pressing this button would lower the unit symbol to the bottom of the stack if one or more units are located beneath the visible one. The bottom pushbutton is labeled with the identification of the airbase. Pressing this button brings up a status window for the airbase. A complete listing of the software can be found at the end of this article.

For this sample program, we need to "include" the SAIC bindings and Boeing's Xlib, Xt Intrinsics, and the Motif widgets. To make the program easier to read, we abbreviate (or rename) the X_TOOLKIT_INTRINSICS_OSF package as simply XT. Additionally, we need access to the Verdix package COMMAND_LINE and the predefined Ada package SYSTEM. A few declarations need to be made before getting to the body of the main procedure. First we need declarations for the top level, form and pushbutton widgets. We also need a declaration for the argument list used to specify resources for the widgets.

```
Top_Level_Widget  : XT.Widget;
Airbase_Form      : XT.Widget;
Airbase_Symbol    : XT.Widget;
Airbase_Name      : XT.Widget;
Args              : XT.Arg_List := XT.Null_Arg_List;
```

Several declarations are needed to create the graphics context (GC) for the airbase symbol. The specific declarations, however, depend on the method actually used to create the GC. Oliver Jones presented one method for creating GC's in the September/October issue of *The X Journal*. An alternate method, presented here, utilizes the *Create_Pixmap_From_Bitmap_Data* command to process the data created by the Bitmap editor. To use this function, the pixmap data must be located in the Ada program. Thus, we need declarations for the pixmap that is to be created along with its width, height and bit data. The bit data must be edited so that it is in Ada hexadecimal format. For the sake of brevity, some of the bit data has been omitted.

```
Airbase_Pixmap     : X_Windows.Pixmap;
Airbase_Width      : X_Windows.Pixels := 50;
Airbase_Height     : X_Windows.Pixels := 30;
Airbase_Bit_Array : X_Windows.Bit_Data_Array( 1..210 ) := (
                        16#00#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#80#, 16#0f#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#80#, 16#08#,
                        16#08#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#c0#, 16#18#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#88#, 16#88#, 16#00#, 16#00#,
                        .
                        .
                        .
                        16#00#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00# );
Airbase_Bit_Data   : X_Windows.Bits :=
                      new X_Windows.Bit_Data_Array'( Airbase_Bit_Array );
```

When we create the pixmap, we need to specify foreground and background colors for the symbol. While theses values can be read in from a resource file, they are hardcoded in this example.

```
Airbase_Fg_Color   : X_Windows.Pixel := 12;     -- Blue
Airbase_Bg_Color   : X_Windows.Pixel := 0;      -- White
```

The function to create the pixmap is an Xlib (SAIC) subroutine that cannot deal with widgets directly. Thus, we need to obtain display, window, and screen depth data structures and resource identifiers for the parent widget. The following variables are used to hold this information.

```
Boeing_Display_Id  : XLIB.Display_Pointer;
SAIC_Display_Id    : X_Windows.Display;
SAIC_Root_Window   : X_Windows.Drawable;
SAIC_Default_Depth : X_Windows.Depth_Type;
```

As shown above, two variables are needed for the display structure because of different types used by the SAIC and Boeing bindings. The *Xt_Display* function returns a type defined by the Boeing bindings. Because this value is needed by the SAIC routine to create the pixmap, the value must be converted to an SAIC type. To accomplish this type conversion, we create an instance of the generic Ada unchecked conversion function. A similar conversion routine is needed for the drawable structure. One last conversion routine converts the pixmap to an address. Its purpose will be explained further as it is used.

```
function Display_Id_From_Xt_Display is new Unchecked_Conversion
                                   ( Source => XLIB.Display_Pointer,
                                     Target => X_Windows.Display );
function Drawable_Id_From_Xt_Drawable is new Unchecked_Conversion
                                   ( Source => XLIB.Drawable,
                                     Target => X_Windows.Drawable );
function System_Address_From_X_Windows_Pixmap is new Unchecked_Conversion
                                   ( Source => X_Windows.Pixmap,
                                     Target => SYSTEM.ADDRESS );
```

As in all X applications, the first thing the program must do is establish the connection with the X server. We pass in the number and values of the command line arguments provided by the package COMMAND_LINE.

```
Top_Level_Widget := Xt_Initialize( "saber", "Saber",
                                   XT.Null_Option_List,
                                   Command_Line.Argc,
                                   Command_Line.Argv );
```

Once the top level widget is initialized, we can call *Xt_Display* to get the Boeing version of the display id. This value is then converted to the type needed by the SAIC bindings using the unchecked conversion function defined earlier.

```
Boeing_Display_Id  := XT.Xt_Display( Top_Level_Widget );
SAIC_Display_Id    := Display_Id_From_Xt_Display( Boeing_Display_Id );
```

We also need the root window and default depth corresponding to the display id. The SAIC code contains bindings to the Xlib functions *Root_Window* and *Default_Depth*. Unfortunately, these functions cannot be used. Since the connection with the X server was made using the Boeing code and not through the SAIC bindings, several of the internal SAIC data structures were not initialized. The SAIC bindings to *Root_Window* and *Default_Depth* rely on those data structures to function correctly. To counter this problem, bindings for the two Xlib functions were added to the Xlib package of the Boeing bindings. These new bindings return values that are of types specified in the Boeing software. The values returned must, therefore, be converted to the types needed by the SAIC code.

```
SAIC_Root_Window   := Drawable_Id_From_Xt_Drawable(
                          XLIB.Root_Window( Boeing_Display_Id, 0 ) );
SAIC_Default_Depth := X_Windows.Depth_Type(
                          XLIB.Default_Depth( Boeing_Display_Id, 0 ) );
```

With the basic initialization complete, the airbase pixmap can be created by calling the *Create_Pixmap_From_Bitmap_Data* function.

```
Airbase_Pixmap := X_Windows.Graphic_Output.Create_Pixmap_From_Bitmap_Data(
                                                SAIC_Display_Id,
                                                SAIC_Root_Window,
                                                Airbase_Bit_Data,
                                                Airbase_Width,
                                                Airbase_Height,
                                                Airbase_Fg_Color,
                                                Airbase_Bg_Color,
                                                SAIC_Default_Depth );
```

Creation of the airbase form begins with the initialization of the argument list. The Boeing bindings provide a macro to create an argument list of any size. The only constraint is that the size must be greater than or equal to the number of resources to be set. Resources are then set in a manner very similar to the format of normal C code. The Boeing bindings automatically take care of incrementing the counter for the actual number of arguments. The actual creation of the form widget is accomplished through a call to *Xm_Create_Form* as defined in the Boeing software.

```
Xt_Make_Arg_List( SIZE => 2, ARGS => Args );
Xt_Set_Arg( Args, XmN_width, 60 );
Xt_Set_Arg( Args, XmN_height, 60 );
Airbase_Form := Xm_Create_Form( Top_Level_Widget, "form", Args );
```

After creating the form widget, the memory used by the argument list is freed and a new list is created for the next widget. This widget is a Pushbutton widget that shows the name of the airbase. It is attached to the bottom and sides of the Form widget just created.

```
Xt_Clear_Arg_List( Args );
Xt_Make_Arg_List( SIZE => 5, ARGS => Args );
Xt_Set_Arg( Args, XmN_left_Attachment, Xm_ATTACH_FORM );
Xt_Set_Arg( Args, XmN_right_Attachment, Xm_ATTACH_FORM );
Xt_Set_Arg( Args, XmN_bottom_Attachment, Xm_ATTACH_FORM );
Airbase_Name := Xm_Create_Push_Button( Airbase_Form, "Alpha", Args);
```

When the airbase name pushbutton is pressed, we want to create a status window to show relevant information about the base. For this example, we use a stubbed procedure "Show_Airbase_Status". When adding the callback, we use the Ada *address* attribute to obtain the address of the callback procedure. Since there is no client data for this procedure, we pass in *XT.Null_Address*.

```
Xt_Add_Callback( Airbase_Name,
                 XmN_activate_Callback,
                 Show_Airbase_Status'address,
                 XT.Null_Address );
```

The airbase symbol pushbutton is then added just above the name. It is created in much the same fashion as the name button with a couple of exceptions. First, it is attached to the top and sides of the form widget. Also, it is explicitly attached to the name pushbutton on the bottom. The second major difference is that instead of a text string, we wish to display the previously created airbase pixmap on the pushbutton. This is accomplished by using the *XmN_label_Pixmap* and *XmN_label_Type* resources. Since the Boeing bindings do not allow an SAIC pixmap to be passed as a parameter, we must obtain the address of the pixmap using one of our conversion routines.

```
Xt_Clear_Arg_List( Args );
Xt_Make_Arg_List( SIZE => 8, ARGS => Args );
Xt_Set_Arg( Args, XmN_top_Attachment, Xm_ATTACH_FORM );
Xt_Set_Arg( Args, XmN_left_Attachment, Xm_ATTACH_FORM );
Xt_Set_Arg( Args, XmN_right_Attachment, Xm_ATTACH_FORM );
Xt_Set_Arg( Args, XmN_bottom_Attachment, Xm_ATTACH_FORM );
Xt_Set_Arg( Args, XmN_bottom_Widget, Airbase_Name );
Xt_Set_Arg( Args, XmN_label_Pixmap,
            System_Address_From_X_Windows_Pixmap( Airbase_Pixmap ) );
Xt_Set_Arg( Args, XmN_label_Type, Xm_PIXMAP );
Airbase_Symbol := Xm_Create_Push_Button( Airbase_Form, "Airbase", Args);
Xt_Clear_Arg_List( Args );
```

If there are several units in a given location, we want to rotate the units when the symbol pushbutton is pressed, displaying the next unit symbol. We use a stubbed procedure "Lower_Base_Symbol" to perform this function. As for the previous callback, there is no client data.

```
Xt_Add_Callback( Airbase_Symbol,
                 XmN_activate_Callback,
                 Lower_Base_Symbol'address,
                 XT.Null_Address );
```

Now that all of the widgets are created, they can be managed individually. (They could also be managed collectively using the Boeing binding to *Xt_Manage_Children*).

```
Xt_Manage_Child( Airbase_Symbol );
Xt_Manage_Child( Airbase_Name );
Xt_Manage_Child( Airbase_Form );
```

The only remaining tasks are to realize the top level widget and enter the main loop to process events.

```
Xt_Realize_Widget( Top_Level_Widget );
Xt_Main_Loop;
```

4 Conclusion

The Boeing bindings to Xt provide Ada user interface programmers with almost all of the capabilities of their C counterparts. Although the bindings do not provide the full functionality of the Xt Intrinsics and Motif, they are complete enough for all but the most complicated interfaces. Furthermore, the bindings contain sufficient examples to assist in the generation of any missing bindings.

Greater flexibility in user interface design can be obtained by combining the Boeing bindings with those developed by SAIC. As the example in this article showed, some type conversion is needed when passing Xt generated values to SAIC's Xlib routines. However, this type conversion is not needed for all of the SAIC routines.

In summary, the combination of the two bindings finally provides the Ada programmer with the capability to construct professional user interfaces. These interfaces have the look and feel that today's users have come to expect.

Acknowledgements

The research for this paper was supported by a grant from the Air Force Wargaming Center, AU CADRE/WG, Maxwell AFB, AL, 36112.

References

[1] Hyland, Stephen J. "Ada Bindings for Xlib," *The X Journal*, *1*(1) (September/October 1991).

[2] Hyland, Stephen J. and Mark A. Nelson. "Ada Bindings to the X Window System." Ada computer software source code, 1987.

[3] Jones, E. J. "Ada Bindings to the Xt Intrinsics and Motif Widget Set." Ada computer software source code, 1991.

[4] Klabunde, Capt Gary W. *An Animated Graphical Postprocessor for the Saber Wargame*. MS thesis, AFIT/GCS/ENG/91D-10, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.

[5] Klabunde, Gary W. and Mark A. Roth. *Ada Interfaces to the X Window System*. Technical Report AFIT/EN-TR-91-9, Wright-Patterson AFB OH: School of Engineering, Air Force Institute of Technology (AU), December 1991.

Author's Biographies

GARY W. KLABUNDE is a systems analyst/programmer with 12 months experience in writing Ada user interfaces using the X Window System. MARK A. ROTH is an associate professor of computer systems in the School of Engineering at the Air Force Institute of Technology. His current research interests include wargaming simulation and database management systems. Both are currently on active duty for the United States Air Force. Current addresses:

Capt Gary W. Klabunde
SCCC/SOSW
Offut AFB, NE

Maj Mark A. Roth
AFIT/ENG
Wright-Patterson AFB, OH 45433-6583
mroth@afit.af.mil

Appendix: Airbase Symbol Source Code

```
with XM_WIDGET_SET,             -- Boeing bindings to Motif v1.0
     X_TOOLKIT_INTRINSICS_OSF,  -- Boeing bindings to Xt Intrinsics
     XLIB,                      -- Boeing bindings to Xlib
     X_WINDOWS,                 -- SAIC bindings to Xlib
     COMMAND_LINE,              -- Verdix Ada package
     SYSTEM,                    -- Predefined Ada package
     UNCHECKED_CONVERSION;      -- Predefined Ada package

use X_TOOLKIT_INTRINSICS_OSF,
    XM_WIDGET_SET;


procedure Ada_Example is

   package XT renames X_TOOLKIT_INTRINSICS_OSF;

                              -- Boeing Variables
   Boeing_Display_Id  : XLIB.Display_Pointer;
   Top_Level_Widget   : XT.Widget;
   Airbase_Form       : XT.Widget;
   Airbase_Symbol     : XT.Widget;
   Airbase_Name       : XT.Widget;
   Args               : XT.Arg_List := XT.Null_Arg_List;

                              -- SAIC Variables
   SAIC_Display_Id    : X_Windows.Display;
   SAIC_Root_Window   : X_Windows.Drawable;
   SAIC_Default_Depth : X_Windows.Depth_Type;

   Airbase_Pixmap     : X_Windows.Pixmap;
   Airbase_Width      : X_Windows.Pixels := 50;
   Airbase_Height     : X_Windows.Pixels := 50;
   Airbase_Bit_Array  : X_Windows.Bit_Data_Array( 1..210 ) := (
                        16#00#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#80#, 16#0f#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#80#, 16#08#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#00#, 16#80#,
                        16#08#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#c0#, 16#18#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#b0#, 16#68#, 16#00#, 16#00#, 16#00#,
```

```
                        16#00#, 16#00#, 16#88#, 16#88#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#e4#, 16#38#, 16#01#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#fa#, 16#f8#,
                        16#02#, 16#00#, 16#00#, 16#00#, 16#00#, 16#f9#,
                        16#f8#, 16#04#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#fd#, 16#f8#, 16#05#, 16#00#, 16#00#, 16#00#,
                        16#80#, 16#fc#, 16#f8#, 16#09#, 16#00#, 16#00#,
                        16#00#, 16#80#, 16#fe#, 16#f8#, 16#0b#, 16#00#,
                        16#00#, 16#00#, 16#80#, 16#fe#, 16#f8#, 16#0b#,
                        16#00#, 16#00#, 16#00#, 16#80#, 16#fe#, 16#f8#,
                        16#0b#, 16#00#, 16#00#, 16#00#, 16#80#, 16#fe#,
                        16#f8#, 16#0b#, 16#00#, 16#00#, 16#00#, 16#80#,
                        16#fe#, 16#f8#, 16#0b#, 16#00#, 16#00#, 16#00#,
                        16#80#, 16#fc#, 16#f8#, 16#09#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#fd#, 16#f8#, 16#05#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#f9#, 16#f8#, 16#04#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#fa#, 16#f8#,
                        16#02#, 16#00#, 16#00#, 16#00#, 16#00#, 16#e4#,
                        16#38#, 16#01#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#88#, 16#88#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#b0#, 16#68#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#c0#, 16#18#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#80#, 16#08#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#80#, 16#08#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#00#, 16#80#,
                        16#08#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#80#, 16#0f#, 16#00#, 16#00#, 16#00#, 16#00#,
                        16#00#, 16#00#, 16#00#, 16#00#, 16#00#, 16#00# );
   Airbase_Bit_Data    : X_Windows.Bits  :=
new X_Windows.Bit_Data_Array'( Airbase_Bit_Array );

   Airbase_Fg_Color    : X_Windows.Pixels := 12;
   Airbase_Bg_Color    : X_Windows.Pixels := 0;



   function Display_Id_From_Xt_Display is new Unchecked_Conversion
                                      ( Source => XLIB.Display_Pointer,
                                        Target => X_Windows.Display );
   function Drawable_Id_From_Xt_Drawable is new Unchecked_Conversion
                                      ( Source => XLIB.Drawable,
                                        Target => X_Windows.Drawable );
   function System_Address_From_X_Windows_Pixmap is new Unchecked_Conversion
                                      ( Source => X_Windows.Pixmap,
                                        Target => SYSTEM.ADDRESS );

   procedure Lower_Base_Symbol( Symbol_Widget  : in XT.Widget ) is
   begin
      null;
   end Lower_Base_Symbol;
```

```
   procedure Show_Airbase_Status( Name_Widget : in XT.Widget ) is
   begin
      null;
   end Show_Airbase_Status;

begin

                              -- Establish a connection with the X server
   Top_Level_Widget := XT.Xt_Initialize( "board", "Board",
                                         XT.Null_Option_List,
                                         COMMAND_LINE.ARGC,
                                         COMMAND_LINE.ARGV );

                              -- Obtain display, window and depth info
   Boeing_Display_Id  := XT.Xt_Display( Top_Level_Widget );
   SAIC_Display_Id    := Display_Id_From_Xt_Display( Boeing_Display_Id );
   SAIC_Root_Window   := Drawable_Id_From_Xt_Drawable(
                            XLIB.Root_Window( Boeing_Display_Id, 0 ) );
   SAIC_Default_Depth := X_Windows.Depth_Type(
                            XLIB.Default_Depth( Boeing_Display_Id, 0 ) );

                              -- Create the pixmap for the symbol pushbutton
   Airbase_Pixmap := X_Windows.Graphic_Output.Create_Pixmap_From_Bitmap_Data(
                                                      SAIC_Display_Id,
                                                      SAIC_Root_Window,
                                                      Airbase_Bit_Data,
                                                      Airbase_Width,
                                                      Airbase_Height,
                                                      Airbase_Fg_Color,
                                                      Airbase_Bg_Color,
                                                      SAIC_Default_Depth );

                              -- Create the form widget to hold pushbuttons
   XT.Xt_Make_Arg_List( SIZE => 2, ARGS => Args );
   XT.Xt_Set_Arg( Args, XmN_width, 60 );
   XT.Xt_Set_Arg( Args, XmN_height, 60 );
   Airbase_Form := Xm_Create_Form( Top_Level_Widget, "form", Args );
   XT.Xt_Clear_Arg_List( Args );

                              -- Create the name pushbutton
   XT.Xt_Make_Arg_List( SIZE => 5, ARGS => Args );
   XT.Xt_Set_Arg( Args, XmN_left_Attachment, Xm_ATTACH_FORM );
   XT.Xt_Set_Arg( Args, XmN_right_Attachment, Xm_ATTACH_FORM );
   XT.Xt_Set_Arg( Args, XmN_bottom_Attachment, Xm_ATTACH_FORM );
   Airbase_Name := Xm_Create_Push_Button( Airbase_Form, "Alpha", Args);
   XT.Xt_Clear_Arg_List( Args );

   XT.Xt_Add_Callback( Airbase_Name,
                       XmN_activate_Callback,
                       Show_Airbase_Status'address,
                       XT.Null_Address );
```

```
                              -- Create the symbol pushbutton
   XT.Xt_Make_Arg_List( SIZE => 8, ARGS => Args );
   XT.Xt_Set_Arg( Args, XmN_top_Attachment, Xm_ATTACH_FORM );
   XT.Xt_Set_Arg( Args, XmN_left_Attachment, Xm_ATTACH_FORM );
   XT.Xt_Set_Arg( Args, XmN_right_Attachment, Xm_ATTACH_FORM );
   XT.Xt_Set_Arg( Args, XmN_bottom_Attachment, Xm_ATTACH_FORM );
   XT.Xt_Set_Arg( Args, XmN_bottom_Widget, Airbase_Name );
   XT.Xt_Set_Arg( Args, XmN_label_Pixmap,
                  System_Address_From_X_Windows_Pixmap( Airbase_Pixmap ) );
   XT.Xt_Set_Arg( Args, XmN_label_Type, Xm_PIXMAP );
   Airbase_Symbol := Xm_Create_Push_Button( Airbase_Form, "Airbase", Args);
   XT.Xt_Clear_Arg_List( Args );

   XT.Xt_Add_Callback( Airbase_Symbol,
                       XmN_activate_Callback,
                       Lower_Base_Symbol'address,
                       XT.Null_Address );

                              -- Manage the widgets
   XT.Xt_Manage_Child( Airbase_Symbol );
   XT.Xt_Manage_Child( Airbase_Name );
   XT.Xt_Manage_Child( Airbase_Form );

   XT.Xt_Realize_Widget( Top_Level_Widget );

   XT.Xt_Main_Loop;

end Ada_Example;
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 December 1991	Technical Report

4. TITLE AND SUBTITLE
Using Ada Bindings to the X Window System

5. FUNDING NUMBERS

6. AUTHOR(S)
Gary W. Klabunde, Capt, USAF
Mark A. Roth, Maj, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/EN-TR-91-10

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Wargaming Center
AU CADRE/WG
Maxwell AFB AL, 36112-5532

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
In this paper we present a detailed example of how to use the SAIC and Boeing bindings to interface an Ada program to X. We discuss issues involving SAIC/Boeing binding compatibility, pixmap creation and use, connection with the X server, creation of several different widgets, and adding appropriate callback procedures.

14. SUBJECT TERMS
Ada, X Windows, Motif, Ada Bindings

15. NUMBER OF PAGES
16

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL